Entwicklungsprozess

Haake, Jörg ; Schwabe, Gerhard ; Wessner, Martin

Erschienen in: CSCL-Kompendium, von J.M. Haake; G. Schwabe; M. Wesser (Hrsg.), Oldenbourg Verlag 2004, München, S. 294 - 300

Entwicklungsprozess

Jörg M. Haake[1], Gerhard Schwabe[2], Martin Wessner[3]

[1]FernUniversität in Hagen, [2]Universität Zürich,
[3]Fraunhofer IPSI, Darmstadt

1 Socio-Cognitive Engineering

CSCL-Anwendungen können analog zu CSCW-Anwendungen als spezielle Art von Software betrachtet werden. Aufgrund ihrer Eigenheiten lassen sich Methoden der klassischen Softwareerstellung für nicht kooperative Anwendungen nicht unverändert anwenden. So verläuft die Entwicklung von CSCL-Anwendungen nicht als Folge separater Phasen sondern als Folge vieler kurzer Zyklen, in denen die Phasen der klassischen Softwareentwicklung (z.B. Anforderungsanalyse, Spezifikation, Design, Implementierung und Test) verschränkt auftreten. Hierdurch wird erst die Ko-Evolution von Anwender- bzw. Entwicklerverständnis einerseits sowie des Softwaresystems andererseits möglich. Aufgrund der Koevolution von Anwenderverständnis bzw. auch -verhalten und der zu ihrer Unterstützung entwickelten CSCL-Anwendung müssen CSCL-Systeme als sozio-kognitive Systeme (Sharples et al. 2002) gestaltet werden. Sozio-kognitive Systeme umfassen hier sowohl die Aspekte, die die Nutzung des Systems durch einzelne Benutzer betreffen als auch die Aspekte, die die Nutzung des Systems in einer (z.B. arbeitsteiligen) Organisation betreffen. Letzteres ist ein wesentlicher Aspekt von CSCL-Anwendungen. Sharples et al. (2002) schlagen hierfür die Methodologie „Socio-Cognitive Engineering“ vor. Ausgangspunkt dieser Methodologie ist die Analyse der komplexen Interaktionen zwischen Menschen und computer-basierter Technologie mit dem Ziel der Überführung dieser Analyse in benutzbare, nützliche und elegante Technologie in ihrem sozialen Kontext.

Der Ansatz besteht aus zwei Stufen (vgl. Abbildung 1): die Analyse der Aktivitäten und das Design der Technologie. Ausgangspunkt ist die Erhebung der allgemeinen Anforderungen. Bei der Analyse der Aktivitäten wird der Typ der zu unterstützenden Arbeit definiert, die Domäne (z.B. im CSCL eine Vorlesung in linearer Algebra) wird festgelegt und es werden die Randbedingungen (z.B. Ausstattung der Zielgruppe, vorgesehene Didaktik) für das Design erhoben. Daran schließen sich zwei parallele Studien an: eine Feldstudie, die betrachtet, wie Menschen die zu unterstützende Arbeit im normalen Arbeitskontext erledigen, und eine mehr Theorie-basierte Untersuchung der zugrunde liegenden kognitiven und sozialen Prozesse. Die beiden Studien werden dann in einem Aufgabenmodell synthetisiert. Im Aufgabenmodell werden die Interaktionen zwischen Personen, Werkzeugen und Ressourcen beschrieben, und es wird analysiert, wie die Personen ihre Arbeit externalisieren (z.B. in Form von Dokumenten oder Notizen), welche Regeln und Konventionen ihre Aktivitäten beeinflussen, welche Terminologie die Personen verwenden und welche Muster in ihren Diskursen vorkommen.

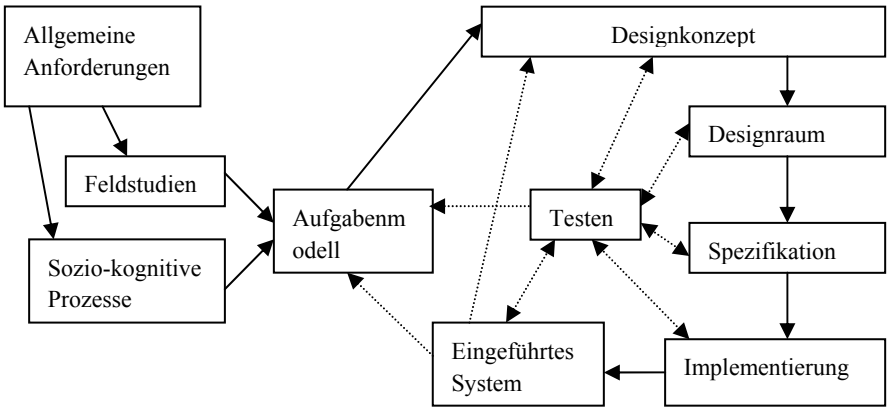

Abbildung 1: Hauptergebnisse und ihre Abfolge beim Designprozess (nach Sharples et al. (2002))

An diese erste Stufe der Analyse schließt sich ein iterativer Designzyklus an. Dieser besteht aus folgenden Aktivitäten: Spezifikation eines Designkonzepts, Erzeugung des dazugehörigen Designraums (d.h. der Menge der möglichen Designalternativen für dieses Konzept), Spezifikation der funktionalen und nicht-funktionalen Aspekte des Systems, Implementierung und Einführung des Systems. Das so entwickelte Gesamtsystem, das über das technologische System (Hardware und Software) und die Dokumentation hinaus auch die empfohlenen Nutzungsmethoden umfasst, soll dann den Anforderungen des Aufgabenmodells genügen. Kontinuierliche Tests und Evaluierung führen zu iterativen Verbesserungszyklen innerhalb des groben Vorgehensmodells.

Für die Durchführung der Methodologie schlagen Sharples et al. (2002) ein Framework aus Bausteinen für das sozio-kognitive Systemdesign vor (siehe Tabelle 1). Es werden vier Prozesse unterstützt: Software Engineering, Task Engineering, Knowledge Engineering und Organizational Engineering. Vier Spalten enthalten Bausteine für diese Prozesse, die sich an den neun Hauptphasen des Entwicklungsprozesses orientieren (Vorschlagen, Erheben, Analysieren, Interpretieren, Design, Implementieren, Integrieren, Evaluieren und Wartung) und aufeinander aufbauen. Jeder Baustein bezeichnet eine Aktivität, zu der mehrere Methoden angewendet werden könnten. Sharples et al. (2002) betonen, dass ihr Framework keineswegs für die reine sequentielle Anwendung gedacht ist, sondern dass die Entwickler in jeder Phase in unterschiedlichem Ausmaß aktiv werden können. Da aber Modifikationen in einer Phase alle davor liegenden Phasen betreffen können, müssen diese erneut betrachtet werden, um die Konsistenz des Gesamtsystems zu garantieren. Aufgrund kontinuierlicher Tests und solcher Änderungen kommt es zu zahlreichen Iterationszyklen während des Entwicklungsprozesses.

Tabelle 1: Framework aus Bausteinen für das sozio-kognitive Systemdesign (nach Sharples et al. (2002))

Phasen	Software Engineering	Task Engineering	Knowlegde Engineering	Organizational Engineering
1. Vorschlagen	Allgemeine Anforderungen			
2. Erheben	Existierende Systeme	Konventionelle Aufgabenstrukturen & Prozesse	Domänenwissen	Organisatorische Strukturen und Abläufe
3. Analysieren	Anforderungen	Aufgaben: Ziele, Objekte, Methoden	Wissen: Konzepte, Fertigkeiten	Arbeitsplatz: Praktiken, Interaktionen
4. Interpretieren	Aufgabenmodell			
5. Design	Algorithmen, Heuristiken	Mensch-Computer Interaktion	Domänenkarte, Benutzermodell	Sozio-technisches System
6. Implementieren	Prototypen, Dokumentation	Interfaces, Kognitive Werkzeuge	Wissensrepräsentation	Kommunikation, vernetzte Ressourcen
7. Integrieren	Protoypisches System			
8. Evaluieren	Debugging	Benutzbarkeit	Conceptual change, skill development	Organisatorische Änderungen
9. Wartung	Installiertes System	Neue Aufgabenstruktur	Angereichertes Wissen	Neue Organisationsstruktur

2 Zyklische Vorgehensmodelle

Betrachtet man CSCL-Anwendungen aus Implementierungssicht als CSCW-Anwendungen, dann lassen sich die Methoden der Softwareentwicklung, die für CSCW-Anwendungen entwickelt wurden, übertragen. Typisch für die Entwicklung von CSCW-Anwendungen ist die Entwicklung einer Folge von Releases, wobei für jedes Release die Rahmenbedingungen und Anforderungen definiert, ein Design entwickelt, implementiert, evaluiert und eingeführt bzw. betrieben werden muss. Hierfür bieten sich evolutionäre Methoden der Softwareerstellung an, z.B. das STEPS Modell (Floyd et al. 1989; 1994; 1997). STEPS definiert einen evolutionären Entwicklungsprozess, in dem eine Folge von „Systemversionen“ (= Releases) als Reaktion auf geänderte Anforderungen der Anwender entwickelt wird. Softwaredesign stellt einen gemeinsamen Lernprozess von Entwicklern und Anwendern dar. Hierzu ist eine intensive Kommunikation zwischen Anwendern und Entwicklern notwendig. Pankoke-Babatz et al. (2001) haben das STEPS-Modell für die Erstellung von CSCW-Anwendungen um die parallele Fortentwicklung von Anforderungsdefinition, Design, Implementierung und Betrieb erweitert.

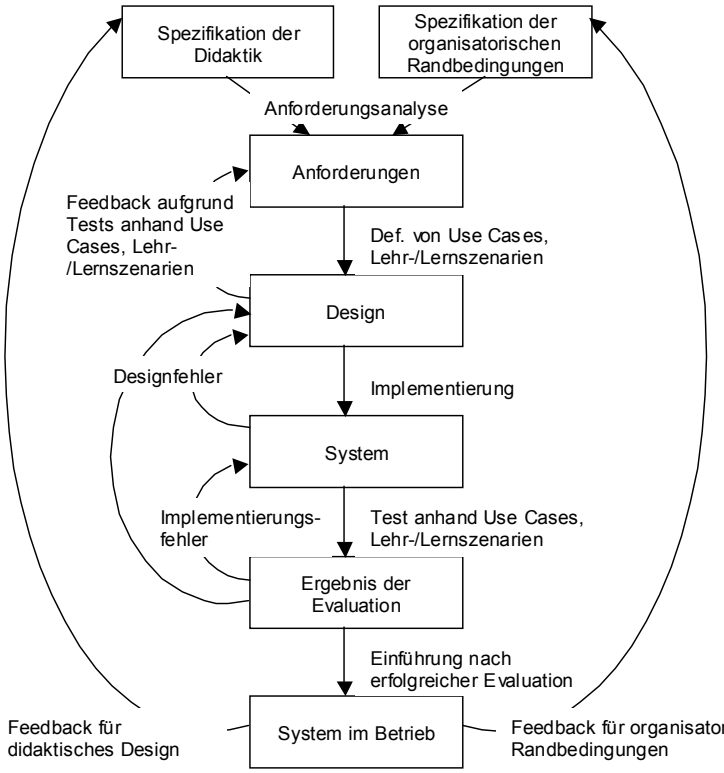

Abbildung 2: Gesamtprozess der Entwicklung und des Betriebs eines CSCL-Systems

Abbildung 2 stellt den Gesamtprozess der Entwicklung und des Betriebs einer CSCL-Anwendung dar, wie er an der FernUniversität in Hagen erfolgreich eingesetzt wird. Dieses Vorgehen kann als Beispiel für eine konkrete Anwendung der Methode von Sharples et al. (2002) sowie des erweiterten STEPS Modells angesehen werden. In der ersten Stufe (Analyse) werden ausgehend von einer Bedarfsanalyse die Anforderungen aus didaktischer und organisatorischer Sicht festgelegt. Die so definierten Anforderungen definieren die Ziele des Systemdesigns. Das Aufgabenmodell wird konkret über die Definition von Anwendungsfällen (Use Cases) und Lehr-/Lernszenarien genauer festgelegt – hierbei spielen Lerntheorien und didaktische Modelle ein wesentliche Rolle bei der Festlegung der sozio-kognitiven Prozesse beim verteilten Lernen. In der zweiten Stufe wird das System iterativ realisiert. Das Systemdesign gibt an, wie diese Anforderungen von den Systemkomponenten mit Hilfe der Benutzungsoberfläche und der zu entwickelnden Systemfunktionen erfüllt werden sollen. Anhand der Anwendungsfälle (Use Cases) und Lehr-/Lernszenarien (d.h. Aufgabenmodelle) kann das Design mit den späteren Anwendern überprüft und ggf. korrigiert werden. Die Implementierung des Designs führt dann zu einem System. Werden bei der Implementierung Designfehler festgestellt, so müssen diese in einem Re-Design korrigiert werden. Das lauffähige System kann dann in einem Evaluationsschritt anhand der Aufgabenmodelle (Anwendungsfälle und Lehr-/Lernszenarien) getestet und ggf. korrigiert werden. Bei positiver Evaluierung kann das System eingeführt werden und in Betrieb gehen. Die Erfahrungen im Betrieb können wiederum als Feedback in die Anpassung der didaktischen und organisatorischen Anforderungen und der Aufgabenmodelle eingehen und im nächsten Entwicklungszyklus wirksam werden.

Dieses Modell des Gesamtprozesses der Entwicklung einer CSCL-Anwendung umfasst die wesentlichen Aktivitäten des STEPS Modells: Systemgestaltung (hier: Spezifikation der Randbedingungen und Anforderungen), Systemspezifikation (hier: Design),

Softwarerealisierung (hier: Implementierung des Systems) und Einsatz (hier: Einführung, Betrieb). Auf die explizite Bezeichnung von Systemversionen sowie der Projektetablierung wurde aus Gründen der Übersichtlichkeit verzichtet.

Eine Konkretisierung von STEPS betrifft die explizite, mit Design und Implementation verschränkte, Evaluation (Pfister & Wessner 2000) und die daraus resultierenden Zyklen im Modell sowie die Unterstützung des sozio-technischen Designprozesses. Die explizite, verschränkte Evaluation wird notwendig, da für die Evaluation neuartiger Systemtypen wie CSCL-Anwendungen es in der Regel unmöglich ist, von Beginn an einen begründeten Kriterienkatalog aufzustellen. Pfister & Wessner (2000) argumentieren deshalb für eine verschränkte formative Evaluation auf zwei Ebenen: zum einen auf der Ebene der Systementwicklung, zum anderen auf der Ebene der Evaluationskriterien. Die Systementwicklung muss iterativ den Resultaten des formativen Evaluationsprozesses angepasst werden. Gleichzeitig müssen auch die Evaluationskriterien selbst iterativ den Erkenntnissen und Randbedingungen, die erst im Zuge der Systementwicklung deutlich werden, angepasst werden. Nach Pfister & Wessner (2000) wird formative Evaluation hier also nicht nur als kontinuierliche Evaluation und iteratives Redesign des Systems, sondern ebenso als kontinuierliche Reformulierung des Kriterienkatalogs selbst verstanden. Zwar bleibt auf oberster Ebene der Gegenstand der Evaluation konstant, die konkrete Kriterienhierarchie einschließlich der Operationalisierungen ist jedoch hochgradig mit dem aktuellen Stand der Systementwicklung verschränkt. Evaluation muss also selbst als ein kooperativer Lernprozess begriffen werden!

Wie oben ausgeführt erfordern Design und Implementierung von CSCL-Systemen viele Zyklen, da Anwender ihr Verständnis im Entwicklungsprozess vertiefen müssen. Zentral ist für diesen Prozess daher das intensive und direkte Einbeziehen der Anwender. Um eine konkrete und intensive Kommunikation zwischen Anwendern und Entwicklern zu fördern, ist der frühzeitig Einsatz von Prototypen für Design (z.B. in Papierform) und Implementierung (als Systemprototyp) förderlich.

3 Einfluss der organisatorischen Bedeutung von CSCL

Die bisherigen Ausführungen gehen von dem derzeit typischen Wissensstand zu CSCL aus: Organisationen experimentieren mit CSCL, um in Pilotprojekten Erfahrungen zu sammeln. Deshalb sind ein enger Bezug zum Anwender und das Sammeln von Erfahrungen während des Entwicklungsprozesses von großer Bedeutung. Diese Annahme kann für einzelne, professionelle Anbieter schon heute nicht mehr zu treffen und die Zahl der Erfahrenen nimmt mit der Diffusion der Technologie zu. Im Folgenden wird deshalb die vorgestellte Vorgehensweise in einen organisatorischen Kontext gestellt und diskutiert, wie die Vorgehensweise den Zielen und dem Erfahrungsstand der Organisation anzupassen ist.

Aus organisatorischer Sicht hat die Bedeutung einer Anwendung für eine Organisation (z.B. Unternehmen oder Universität) einen großen Einfluss auf den Entwicklungsprozess. Ward & Peppard (2003) unterscheiden hier in Anlehnung an das Produktportfolio zwischen strategischen Anwendungen, operativ kritischen Anwendungen, High-Potential Anwendungen und Unterstützungsanwendungen (vgl. Abb. 3). Applegate et al. (2003) machen die gewählte Entwicklungsvorgehensweise von den Erfahrungen der Anwenderorganisation mit

der Technologie und den mit der Einführung verbundenen organisatorischen Änderungen abhängig.

In Zukunft	**Strategisch**	**High Potential**
	Anwendungen, die kritisch für die Umsetzung der Unternehmensstrategie sind	Anwendungen, die möglicherweise für den zukünftigen Erfolg eine wichtige Rolle spielen
Heute	Anwendungen, von denen die Organisation zur Zeit erfolgsabhängig ist	Nützliche Anwendungen, die aber nicht erfolgskritisch sind
	operativ kritisch	**Unterstützung**
	Hohe Bedeutung	Niedrige Bedeutung

Abbildung 3: Derzeitige und zukünftige Bedeutung von Anwendungen

Für die meisten Organisationen sind CSCL-Anwendungen derzeit High-Potential Anwendungen oder Unterstützungsanwendungen. Im Fall von High-Potential Anwendungen zielen CSCL-Projekte darauf ab, den Nutzen von CSCL kennen zu lernen und die in diesem Beitrag vorgestellte Methode ist in ihrer Gesamtheit anzuwenden. Wegen seines hohen Risikos sollte das CSCL-Projekt von anderen Anwendungen (insbesondere solche mit hoher Bedeutung) isoliert werden (z.B. indem Daten nicht integriert werden), damit bei einem Fehlschlag wichtige Systeme nicht beeinträchtigt werden.

Ist eine CSCL-Anwendung nützlich, aber nicht geschäftsentscheidend, so hat sie für eine Organisation einen nur unterstützenden Charakter. In diesem Fall kommt es darauf an, zu einer möglichst kostengünstigen Lösung zu gelangen. Dies ist in der Regel durch den Einsatz von Standardsoftware zu erreichen. Damit reduziert sich der Software-Engineering Teil der Entwicklung (vgl. Tabelle 1) auf das Anpassen und „Customizing“ von Standardsoftware. Auch die Anforderungen werden - soweit möglich - nicht von Grund auf neu definiert, sondern durch die Anpassung von Referenzmodellen an die individuellen Bedürfnisse einer Organisation erhoben. Eine Nutzerbeteiligung ist auch in diesem Fall sinnvoll; es ist sogar denkbar, dass die Nutzer die Implementierung des Systems weitgehend selbst übernehmen.

Eine strategische Bedeutung kann CSCL beispielsweise für Fernuniversitäten haben, d.h. diese Organisationen erwarten sich einen Wettbewerbsvorteil davon, dass die dort eingeschriebenen Studierenden ihre Lernziele mit Hilfe von kooperativen Medien erreichen. Erste Erfahrungen sollten in High-Potential-Projekten gesammelt werden, aber das darauf folgende strategische Projekt ist dann auch ein Lernprozess, der wiederum eine iterative Vorgehensweise erfordert. Können High-Potential Projekte noch stark Technologie-getrieben sein, ist bei strategischen Projekten eine Leitung durch einen Anwender (d.h. für eine Fernuniversität beispielsweise durch einen Dozenten) wesentlich. Die große Bedeutung des Projekts erfordert einen starken und mit aktiven Personen besetzten Lenkungsausschuss. Aus dem gleichen Grund ist auch ein eigener Aktivitätenstrang zum Risikomanagement geboten. Die Risiken sollten, in Anlehnung an das Spiralmodell von Boehm (1986), für jede Iteration bei der Entwicklung der CSCL-Anwendung in einer eigenen Phase beurteilt werden.

Die Mehrzahl der in diesem Buch diskutierten CSCL-Anwendungen hat noch nicht den Status einer operativ-kritischen Anwendung erreicht, d.h. es hängt das derzeitige „Geschäft“ noch nicht davon ab, dass CSCL-Systeme (jenseits einfacher E-Mail-Systeme oder Newsgroups) im Dauerbetrieb funktionieren. Wenn CSCL Anwendungen operativ-kritisch sind, dann haben Neuprojekte in diesem Umfeld das Ziel, die derzeitigen Systeme zu verbessern und man kann davon ausgehen, dass bei Betreibern und Anwendern ausreichend Erfahrungen im Umgang mit diesen Systemen vorliegen. Deshalb kann dann auf eine Einbeziehung der Anwender in den laufenden Entwicklungsprozess zu Gunsten einer gründlichen Anforderungserhebung zu Beginn verzichtet werden. Der eigentliche Schwerpunkt der Aktivitäten liegt dann auf einer guten Ingenieur-Leistung bei der Konstruktion und Implementierung der Lösung. Hierzu gibt es ausreichende Gestaltungshinweise im klassischen Software-Engineering. Eine eigene Vorgehensweise für die Entwicklung von CSCL-Systemen ist dann nicht mehr notwendig.